Projet :

Reconnaissance faciale pour remplacer la feuille d’éamrgement

Développment d’une solution ambarqué basé sur l’IA

Stratégie :

Recherche des méthodes de reconnaissance faciales et des models de l’IA

Faire une étude théorique comparative

Faire une étude éxpérimentale comparative avec des résultats

Collection de la base de donnée et génération et faire le développement, le test et la validation de modele

Etudes des différents cas possibles et développement des solutions pour des différents cas possible

Implémentation de la solution dans un système embarqués

Amélioration de programme en terme de perforamnce et robustesse

Cahier des charges :

Méthode de travail :

On a répartie le travail sur deux, Arslane est responsable a la partie implémentation du programme et la partie de développement des solution de la livraison des résultats.

Iheb était résponsable pour le développement et la validation du code IA.

J’ai commencé par la réalisation d’un vérité terrain pour ma base de donnée.

J’ai développé un logigramme pour la solution que je veux proposer

Je développe, je teste, je valide ou je modifie et je teste encore une fois et je valide ou je modifie.

Etudes des cas possible et développement de solutions pour différents cas.

**DeepFace** is a Python library for deep learning-based face recognition. It provides a simple interface for common facial recognition tasks, including detecting faces, analyzing facial attributes, and matching faces. OpenCV (Open Source Computer Vision Library) plays a crucial role in the pre-processing steps before applying deep learning models for facial analysis.

**Key Uses of OpenCV in DeepFace:**

1. **Face Detection**:
   * OpenCV provides several pre-trained classifiers for face detection, such as Haar Cascades and DNN-based face detectors. These are used to detect and localize faces in images or video frames.
   * The library uses functions like **cv2.CascadeClassifier** for loading Haar cascades and **cv2.dnn.readNetFromCaffe** for loading models for DNN-based face detection.
2. **Image Preprocessing**:
   * OpenCV is used for various image preprocessing tasks, such as resizing, cropping, and converting images between different color spaces.
   * Functions like **cv2.resize**, **cv2.cvtColor**, and **cv2.equalizeHist** are commonly used to prepare images before feeding them into deep learning models.
3. **Data Augmentation**:
   * For training models, OpenCV helps in augmenting the dataset by performing transformations like rotation, scaling, and flipping. This increases the variability of the training data and helps in generalizing the models better.

**VGG-Face Model**

**VGG-Face** is a deep convolutional neural network model developed by the Visual Geometry Group (VGG) at the University of Oxford for face recognition tasks. It is trained on a large dataset of face images and achieves high accuracy in recognizing and verifying faces.

**Key Characteristics of VGG-Face:**

1. **Architecture**:
   * VGG-Face is based on the VGG-16 architecture, which consists of 16 layers: 13 convolutional layers followed by 3 fully connected layers.
   * The convolutional layers use small receptive fields (3x3 kernels) and are stacked with increasing depth to learn hierarchical representations of the input images.
2. **Training**:
   * The model was trained on a large dataset consisting of over 2.6 million face images from more than 2,600 individuals.
   * The training process involved optimizing the weights of the network to minimize the classification loss, allowing the model to learn robust facial features.
3. **Feature Extraction**:
   * After training, the VGG-Face model can be used to extract high-dimensional feature vectors (embeddings) from face images.
   * These embeddings capture the unique characteristics of faces and can be used for various tasks such as face verification (comparing two faces) and face recognition (identifying a person from a database).

**Integration of OpenCV and VGG-Face in DeepFace:**

In the DeepFace library, OpenCV and VGG-Face are integrated to provide an end-to-end solution for face recognition. Here's how they work together:

1. **Face Detection with OpenCV**:
   * OpenCV is used to detect faces in an input image. The detected face regions are then cropped and preprocessed.
2. **Face Embedding with VGG-Face**:
   * The preprocessed face images are fed into the VGG-Face model to obtain face embeddings.
   * These embeddings are high-dimensional vectors that represent the unique features of the faces.
3. **Face Matching**:
   * To compare faces, the embeddings are used to compute similarity scores (e.g., cosine similarity). If the similarity score is above a certain threshold, the faces are considered to match.
4. **Attribute Analysis**:
   * Besides recognition, the embeddings can also be used for attribute analysis, such as predicting the age, gender, and emotion of the detected faces.

<https://www.datacamp.com/tutorial/face-detection-python-opencv>

<https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>

<https://docs.opencv.org/4.x/d0/dd4/tutorial_dnn_face.html>

Rapid Object Detection using a Boosted Cascade of Simple Features (paper of cascade classifier)

<https://www.geeksforgeeks.org/face-detection-using-cascade-classifier-using-opencv-python/>

Explication de notre projet, qu’est ce qu’il fait, c’est quoi son utilité

Démonstration de la réalisation de projet :

Une petite explication de la relation de la partie software avec la partie hardware.

* Partie développement software :

- Partie de la detection des visages : sortir les position des visage de chaque image en utilisant les méthode de traitement d’image et de machine learning développé en 2001 qui s’appelle cascade classifier et entrainé par opencv donc la méthode c’est de opencv. (integral image et haar pour la detection des caractéristiques), (AdaBoost classifier pour la selection d’un petit nombre de caractéristques car Haar nous donnes un grand nombre, on sort les meilleures caractéristiques) , (après on a le cascade classifier pour une classification rapide).

- Partie de la reconnaissance de visage : on utilisé un modèle qui s’appelle VGG-FACE, entrainé par le modele VGG, ce modele est entrainée sur la base de donnée (…..) et son architecture est définie comme le suivant (……)

* Partie Développement Hardware (implémentation dans le système embarqué)

- Partie Hardware : L’installation de la caméra, la préparartion de l’environement de développement en installant tout les packages et l’implémentation de la solution de code dans la carte.

Les tests et les résultats :

Les tests sur les images des acteurs.

Les tests sur des images avec un seul visage.

Les tests sue des images d’une classe avec plusieurs visages. Et pour plusieurs cas.

Les tests de système en temps réel.

Very good explanation of AdaBoost: <https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm>

The idea is simple, it’s train machine learning models and extract what are the lines of data that gives more error and what are that gives lowest error.

The AdaBoost algorithm:

On prends une base de donnée et on la passe a machine learning model (decision tree par example), ensuite on sort les erreur pour chaque classification et on augmente un cerain poids pour les donnée qui sont mal prédicté, après on fait une probabilité qui dépends des poids et qui prend beaucoup plus les donnée qui sont été mal prédicté, par la suite on reprends cette base de donnée et la rentre dans un autre modele et on fait la meme chose.

Méthode de détection de visage d’OpenCV :

**Définition du projet :**

La conception d’un système de reconnaissance faciale en temps réel pour remplacer la feuille d’émargement dans une salle de cours ou d’examen. Ce système utilise les techniques d’apprentissage profond (Deep Learning) pour identifier les étudiant présents dans la salle à partir de la prise quotidienne des photos par une caméra.

**Cahier de charges :**

Pour la réalisation de ce projet, on a commencé par la définition des objectifs et la faisabilité de ce projet, et on a suivi le cahier des charges suivant :

|  |
| --- |
| **Fonctionnement principale :** |
| Un système qui contient une base de données des étudiant d’une classe. |
| Ce système va prendre une image au minimum une fois chaque 30 minutes des étudiant dans une salle. |
| Le système va traiter cette image et extraire (détecter) des visages dans l’image. |
| Le système va traiter chaque visage et il va faire la reconnaissance faciale de chaque visage. |
| Il va remplir un tableau avec les élèves présents dans la salle. |
| Il va enregistrer toutes les informations tel que l’heure de la prise de l’image, la date et la salle et la classe. |
| Il va envoyer ces informations sur internet. |
| **Cas étranges :** |
| Lorsqu’il y a un élève qui n’est pas de la même classe, le système va détecter cette personne comme inconnue. |
| Le système va enregistrer l’image de visage de la personne inconnue ainsi de l’image initiale complète qui contient tous les visages. |
| Un dossier avec la date du jour et qui contient tous ces fichiers va être envoyer sur internet. |

**Le matériel nécessaire pour la réalisation de ce projet :**

* Une carte Raspberry PI avec son alimentation et une carte mémoire.
* Un module caméra.
* Un pc ordinateur pour le développement de l’application et les tests.
* Un point d’accès internet pour la connexion de la carte avec internet.

**Stratégie de travail :**

Après l’étude de notre cahier de charge, on a décidé de partager le travail en deux parties, une partie de développement informatique de l’application (partie Software) et une partie de développement système embarqué (partie Hardware).

Pour la partie de développement software on a suivi la stratégie suivante :

* On commence par des recherches sur les méthodes qui existe pour la détection et l’extraction des visages et ainsi des modèles de la reconnaissance faciale basée sur le traitement d’image ou des méthodes basées sur l’intelligence artificielle.
* Au même temps on collecte quotidiennement des images pour nos collègues pour l’étude et le teste de notre système.
* Après qu’on trouve des méthodes, on fera des tests sur des images sur internet et ensuite on fera une étude comparative entre les méthodes pour sortir la meilleure méthode qui satisfait notre cahier de charge.
* Après qu’on trouve la méthode ou le modèle IA et le valider, on développe une solution pour notre base de données collecté.
* Après les tests on cherche à valider le modèle sur notre base de données, sinon on doit changer le modèle ou chercher d’autres solutions et refaire les étapes précédentes.

Pour la partie de développement Hardware on a suivi la stratégie suivante :

* On fait la commande du matériel.
* On prépare l’environnement de programmation pour le développement.
* On fait l’implémentation de notre solution dans la carte.
* On fait des tests sur des images.
* On connecte le système avec une caméra et on effectue des tests avec une caméra et en temps réel.
* On développe sur la carte la partie des cas étranges et l’envoie des données.

**Répartition des taches :**

On est deux personnes qui travaille dans ce projet,

Iheb BOUARICHE, va prendre la partie de la recherche et de développement de la solution basée sur l’intelligence artificielle pour la reconnaissance faciale, son rôle et de réalisé un programme qui permet de prendre une image à l’entrée et sortir une liste avec les personnes présentes dans cette image. Il va utiliser les techniques de traitement d’image et de l’apprentissage profond.

Arslane MEGHELLI, va être responsable de développement Hardware, il va réaliser un système embarqué qui va permettre de prendre en temps réel (et à chaque 30 minute maximale) une image et la tourner dans le programme développé par Iheb, et ensuite sortir la liste des étudiants présents et les autres informations pour que par la suite les enregistrer dans un fichier Excel, et à la fin envoyer ces données sur internet.

**Méthodologie de travail :**

Chaque semaine, on a un jour, ou on partage l’avancement de notre travail et les recherches et les solutions et les propositions qu’on a trouvées, ensuite faire des points pour le prochain travail. Et à chaque fois on développe une solution on l’implémente dans la carte pour voir sa faisabilité. On échange les idées et on change la stratégie à chaque fois on se trouve bloqué ou dans une situation compliquée.

**Modification de la stratégie :**

Avant, on a décidé de développer un système pour faire l’apprentissage sur un nombre important des images de nos collègues, après on a déduit que ça ce n’est pas faisable si on a un nouvel étudiant qui rejoindre la classe car :

- Premièrement, ce n’est pas possible de demander à un étudiant un grand nombre de ses photos.

- Deuxièmement, c’est compliqué d’apprendre un modèle de l’IA à chaque fois qu’on a un nouvel étudiant dans la base de données. Sachant que notre système sera implémenté dans une carte Raspberry qui est un petit calculateur et qui est limité en puissance de calcul.

**Solution finale :**

Après des recherches et des tests de plusieurs solutions, on a finalement, validé l’idée de développer un système qui prend une seule photo d’un nouvel élève, et l’ajouter dans la base de données. Ensuite et en mode RUN, à partir de cette image on peut faire la reconnaissance de son visage dans tous les images prise par le système embarqué en temps réel.

**Présentation détaillée :**

1. **Développement software :** Cette partie contient deux parties qui sont très important, la première partie est la détection et l’extraction des visages dans une images qui contient plusieurs visages, et la deuxième partie est la reconnaissance faciale.

Dans ce rapport on va expliquer le fonctionnement de chaque partie, on commence par la première partie :

1. **Extraction et détection des visages :** c’est la localisation des visages dans une image, après une recherche approfondie sur internet, on a trouvé qu’il existe plusieurs méthodes pour l’extraction des visages dans une image qui contient plusieurs visages, notre problème c’était de prendre ces méthodes et tester ses performances pour faire le meilleur choix.

On a pris des images sur internet et on a testé ces méthodes sur différentes images comme vous montre le tableau suivant, on a pris comme paramètres la précision et la vitesse d’exécution (temps d’exécution du programme de l’entrée jusqu’à l’obtention des visages), Voici l’étude comparative des différentes méthodes :

****

**Figure 01 :** l’une des images de test pour le test des méthodes d’extraction des visages.

Voici dans ce tableau les résultats obtenus :

|  |  |
| --- | --- |
| Modèle | Résultats |
| **opencv** |  |
| **Temps de détection** 1.390409231185913 **seconds** |
| **ssd** |  |
| **Temps de détection** 0.13123464584350586 **seconds** |
| **mtcnn** |  |
| **Temps de détection** 2.428948163986206 **seconds** |
| **retinaface** |  |
| **Temps de détection** 8.324228763580322 **seconds** |
| **Yolov8** |  |
| **Temps de détection** 0.33027172088623047 **seconds** |
| **yunet** |  |
| **Temps de détection** 0.06933712959289551 **seconds** |
| **fastmtcnn** |  |
| **Temps de détection** 1.2057855129241943 **seconds** |

**Tableau 01 :** résultats de chaque modèle de l’extraction de visage, ainsi que le temps d’exécution.

**Commentaire :** on remarque que y a plusieurs méthodes qui détecte tous les visages dans l’image, et y en a qui détecte moins et y en a qui se trompe dans la détection.

**Notre choix et explication :** après plusieurs tests on a choisi le modèle d’extraction de visage de « opencv », car ce modèle détecte seulement la position d’un rectangle de chaque visage dans une image, pour les autres méthodes, on voit qu’ils appliquent des transformations sur l’image (tel que l’alignement 3D et la coupure qui ne montre pas tout le visages), ce choix est important pour la partie de la reconnaissance faciale, car pour avoir une bonne précision de reconnaissance on doit garder le maximum de caractéristique d’un visage.

**Principe de fonctionnement de la méthode de l’extraction des visage d’« opencv » :** Maintenant puisque on a choisi la méthode d’« opencv » pour la détection des visages, on a cherché et compris l’article de recherche de cette méthode [], et on va détailler un peu sur son fonctionnement par la suite.

Cette méthode est basée sur des techniques de Machine Learning pour la détection des visages, elle est développée en passant par trois étapes suivantes :

1. **Le calcule de l’image intégrale :** Cette étape consiste à sortir les caractéristiques dans l’image en utilisant des filtres, la méthode de l’image intégrale va nous permettre de calculer ces caractéristiques rapidement en échelle et en position. On va sortir les caractéristiques de HAAR pour cette application.

Les caractéristiques de HAAR : sont des filtres de 2 dimensions et rectangulaire, ces filtres sont utilisés pour l’identification des structures dans une image (tel que les lignes, les textures, etc…), il existe plusieurs types des filtres de HAAR : filtres de deux rectangles, de trois et de quatre rectangles. Ensuite et après l’application de ces filtres sur les images en position et en échelle, on a obtenu un nombre très important de caractéristiques et qui peut être lourd en calcule pour un modèle de classification.

1. **L’algorithme AdaBoost :** cet algorithme est utilisé pour la sélection des caractéristiques les plus importants pour la classification entre une image qui contient un visage et une autre sans visage dans l’image.

AdaBoost est un classificateur qui combine plusieurs classificateurs entrainés pour crier un classificateur puissant.

* Création des Classificateurs : AdaBoost combine ces caractéristiques sélectionnées pour former des classificateurs faibles. Chaque classificateur faible est un modèle simple qui est légèrement meilleur que le hasard pour distinguer entre les visages et les non-visages.

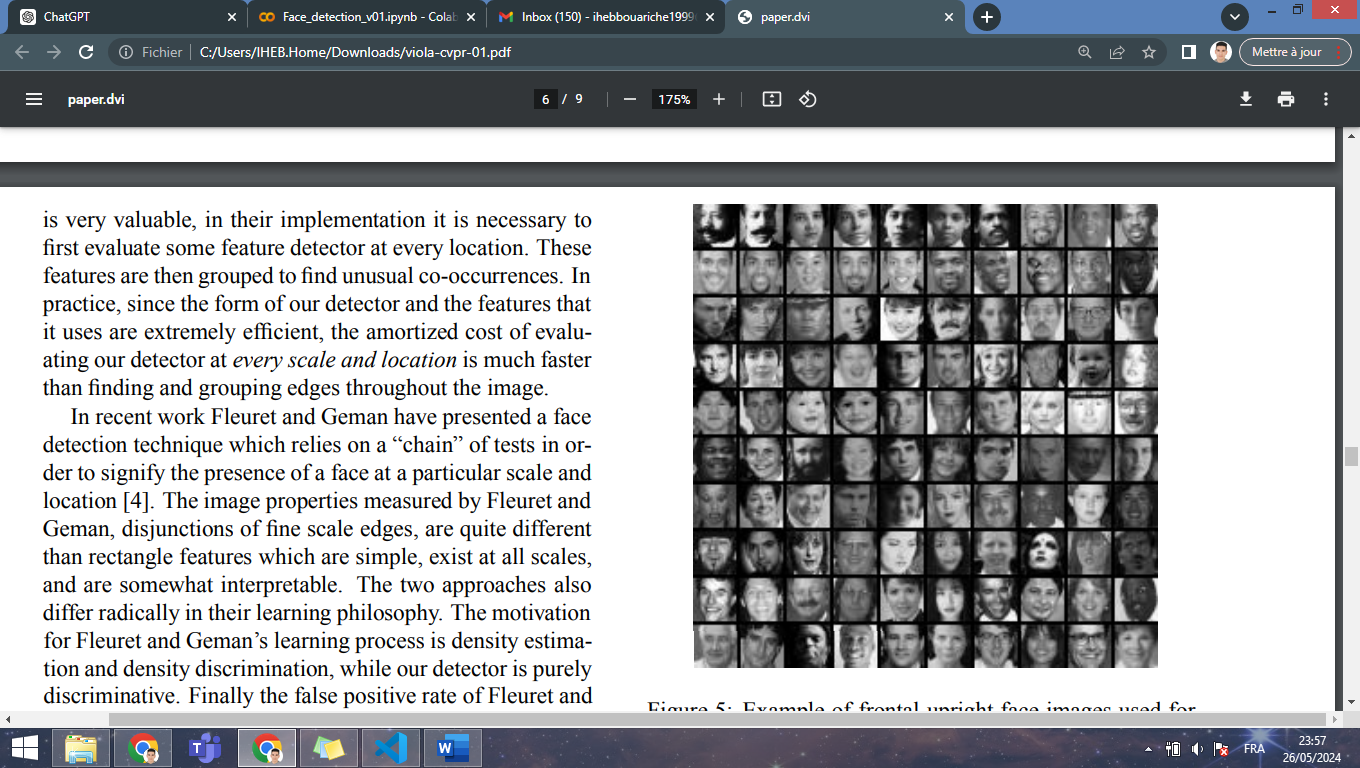
AdaBoost attribue des poids aux exemples d'entraînement, en augmentant les poids des exemples mal classés afin que les classificateurs faibles suivants se concentrent davantage sur ces exemples difficiles.

* Construction d'un Classificateur Fort :

En combinant plusieurs classificateurs faibles pondérés, AdaBoost construit un classificateur fort. Ce classificateur fort est une combinaison linéaire des classificateurs faibles, où chaque classificateur faible contribue en fonction de son poids déterminé par AdaBoost. On dit qu’un classificateur AdaBoost plus fort si il prends plus de caractéristques et donc il a plus de classificateur faibles.

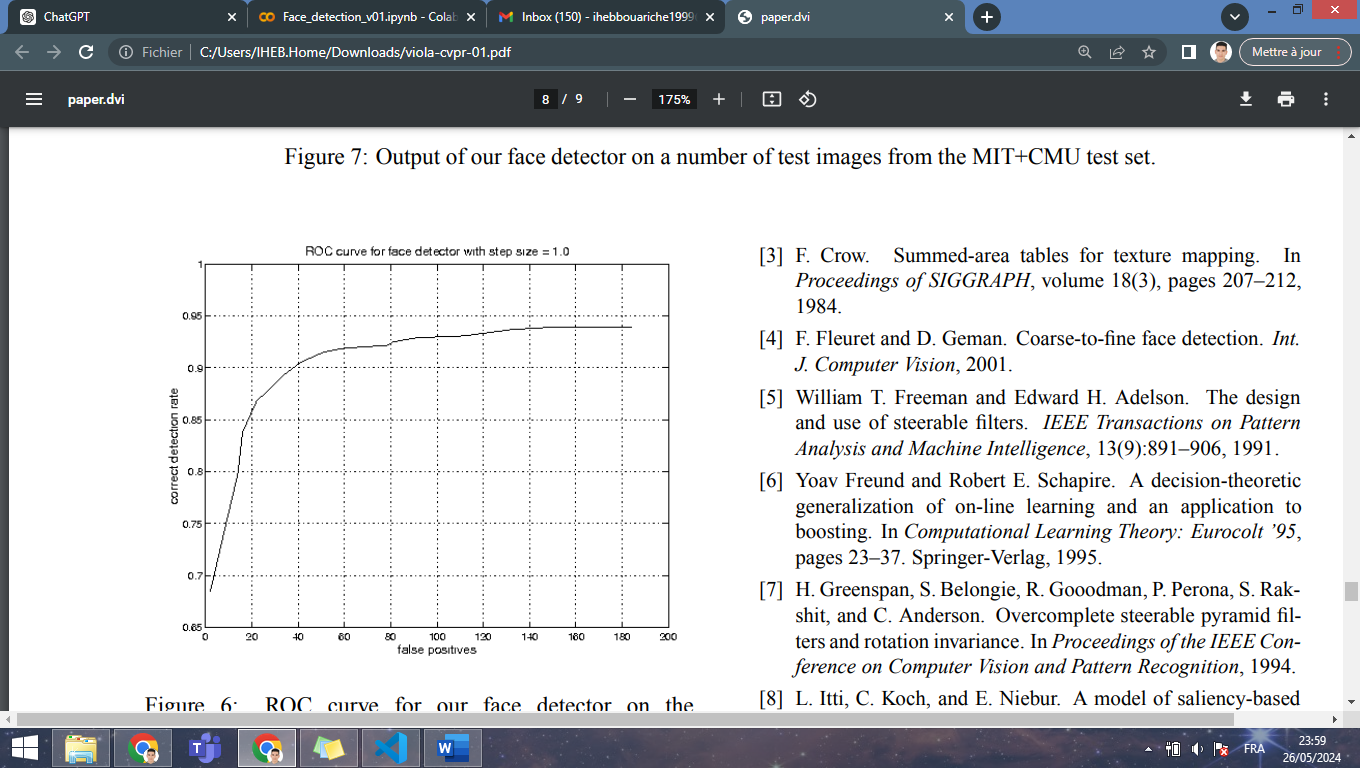
1. **Classification en cascade :** la troisième méthode consiste a réailiser des étages en cascade, chaqye étage contient un classificateur AdaBoost et a chaque fois on monte dans les étage on trouve un modèle plus fort. Cette téchnique est utilisé pour éliminer dans chaque étage les régions qui n’ont pas des visages. Cette technique est utilisé pour avoir une rapidité en calcul. Il faut comprendre que pour avoir un visage dans une région, il faut que cette région passe tous les étages avec succès.

**L’architecture :** Ce modèle utilise 38 couches (étage) de classificateur AdaBoost, entrainée sur une base de données de 4916 image de visages labelisé et avec une résolution de 24 x 24, et 9544 images qui ne contient pas de visages, et ses images ils ont 350 millions petites images de 24x24.



**Figure :** exemple des images des visages utilisé pour l’apprentissage des classificateurs.

**Benchmark :** Voici les résultats dans son article de recherches,



**Interprétation :**

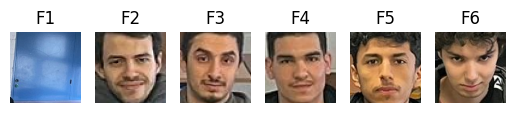
Dans ce graphe on voit dans l’axe des ordonnés les valeurs vraies positive (ou y a un visage) et dans l’axe des absides, des valeurs faux positives (pas de visage), on voit bien que a la fin de l’apprentissage le modèle a atteint un bon rapport de détection des visages dans les images qui égale à peu près 0.94 (d’une autre façon le modèle peut détecter des visages avec un pourcentage 94%).

**Test sur notre base de données :**

Après le test sur une image de notre base de données de nos camarades dans une salle, on a obtenu le résultat suivant :



**Figure 02 :** l’image à l’entrée de système.

****

**Figure 03 :** Les images des visages détectés après l’exécution du code de détection.

**Commentaire :** on remarque bien que le programme a détecté tous les visages qui sont dans l’image à l’entrée (à condition que les visages doivent être positionnés en face la caméra).

Après l’obtention de visage, on va développer une solution qui prend chaque visage et le traiter pour reconnaitre ce visage.

1. **Reconnaissance faciale :** après des recherches sur les méthodes qui peuvent être utilisé pour la reconnaissance faciale, on a trouvé plusieurs façons de faire, tel que :

- Entrainer un classificateur des images, alors qu’on a trouvé que ça nécessite une collecte de donnée importante, donc on a annulé cette méthode.

- La deuxième méthode, est de prendre une image et sortir un vecteur caractéristique d’un visage et ensuite calculer la distance avec d’autre visage, et si la distance est inférieure à un certain seuil donc cela veut dire qu’on a la même personne dans l’image. Cette méthode a bien fonctionné et c’est ce qu’on va utiliser pour notre solution par la suite.

L’avantage est que pour le cas d’un nouvel élève, on prend une seule photo de lui et on peut l’ajouter dans la base de données.

Pour cette solution on a différents modèles de Deep Learning pré-entrainé, et donc on a pris ces modèles et on a testé ces modèles sur différentes images dans notre base de données.

D’une manière simple, le programme va prendre deux images qui ont un visage dans chaque image et rentre ces images l’un après l’autre dans un réseau de neurones profond (modèle de l’IA) et à la sortie on va avoir un vecteur caractéristique de chaque image (en anglais : « embedding vector »), et puis le programme compare la distance entre les deux vecteurs avec un certain seuil. Et si c’est inférieur donc c’est la même personne. Sinon ce n’est pas la même personne, voici le tableau de résultats pour l’un de nos tests :



**Figure 04 :** Deux images de la même personne (avec et sans barbe)

Et voici le tableau avec les différents modèles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Le modèle** | **Distance** | **Temps d’exécution** | **La même personne ?** | **Seuil** |
| VGG-Face | 0.6192805123478301 | 2.4109108448028564 | Oui | 0.68 |
| Facenet | 0.33399397091011995 | 1.6701862812042236 | Oui | 0.4 |
| Facenet512 | 0.32227831143130603 | 1.640101671218872 | Non | 0.3 |
| OpenFace | 0.26733561871553846 | 1.201770305633545 | Non | 0.1 |
| DeepFace | 0.21537111581274848 | 5.225113153457642 | Oui | 0.23 |
| DeepID | 0.047543969387073504 | 1.4541990756988525 | Non | 0.015 |
| ArcFace | 0.4843091708996228 | 1.629307746887207 | Oui | 0.68 |
| SFace | 0.5325555920972895 | 1.0362541675567627 | Oui | 0.593 |

**Commentaire :** On peut voir que les méthodes VGG-Face, Facenet, DeepFace, ArcFace et SFace ont donné la bonne prédiction, alors que les autres ils n’ont pas réussi de détecter que c’est la même personne.

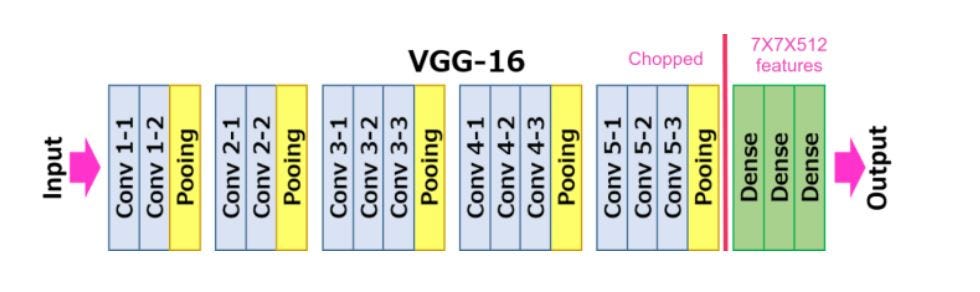
Après plusieurs expériences, on a choisi le modèle VGG-Face (pourquoi, il me faut une étude comparative).

**Explication du modèle VGG-Face :**

Le modèle utilisé pour VGG-Face est basé sur une variante de l'architecture VGG16, qui est un modèle de réseau de neurones convolutif (CNN).

**Son Architecture :**

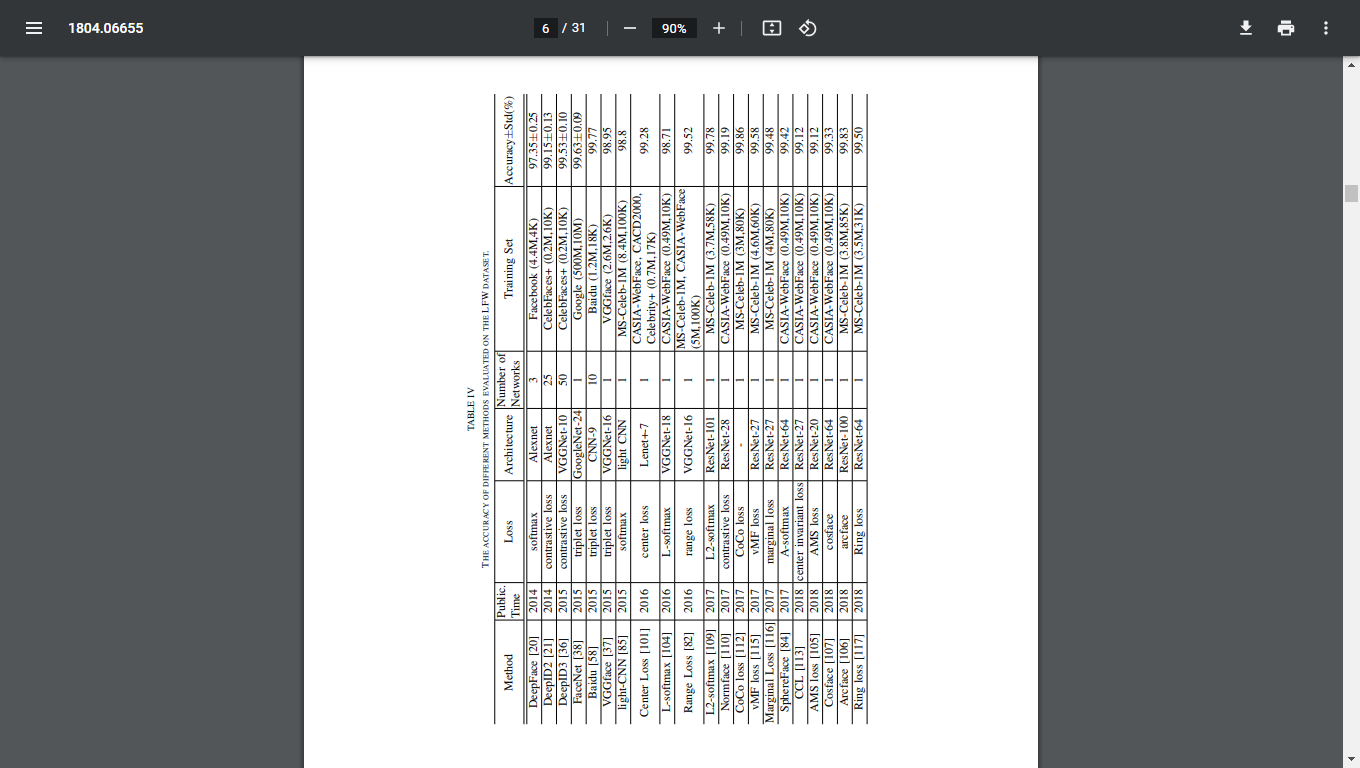
* Le modèle VGG-Face suit l'architecture VGG16, qui se compose de 16 couches.
* Les couches incluent des couches convolutives, des couches entièrement connectées.
* Le réseau se compose typiquement de 13 couches convolutives, 5 couches de max-pooling et 3 couches entièrement connectées.



**Figure :** l’architecture de VGG16.

**Entraînement :**

* Le modèle VGG-Face a été initialement entraîné sur le dataset VGGFace, qui contient 2,6 millions d'images de 2 622 célébrités.
* L'entraînement implique l'utilisation d'un dataset à grande échelle pour capturer une large gamme de variations des visages humains.

**Les résultats comparatifs :**

Ce tableau représente une étude comparative des différents modèles de reconnaissance faciale. On constate que le modèle VGGface contient 16 couches et utilise un seul réseau et il atteint une précision de 98.8%. Par rapport les autres modèles, y en a qui utilisent une taille importante de réseau de neurone, ce qui rend le modèle lent en exécution. Une autre chose, ce modèle est entrainé sur une base de données avec un grand nombre de visages différents (classes) et aussi une grande variation de position, d’âge, d’expression et des conditions d’éclairage, ce qui rend le modèle robuste.

Le modèle VGG-Face a atteint 98.95% de précision avec la base donnée LFW (LFW est une base de données de photographies de visages recueillies à partir d'Internet. Elle a été créée pour aider à la recherche en reconnaissance faciale. La base de données contient plus de 13 000 images de visages de 5 749 personnes différentes. Les images sont prises dans des conditions non contrôlées, incluant des variations en termes de pose, d'expression faciale, d'éclairage et de qualité d'image).

Le modèle VGG-Face a atteint 98.95% de précision avec la base de données YTF (YTF est une base de données vidéo conçue pour l'évaluation de la reconnaissance faciale dans des vidéos. Elle contient des clips vidéo téléchargés depuis YouTube, La base de données comprend 3 425 vidéos de 1 595 personnes. Chaque personne a plusieurs vidéos, offrant une grande variabilité dans les poses, les expressions faciales, les conditions d'éclairage et les arrière-plans).

**Propositions :**

Après la finalisation de ce projet, on peut imaginer tout cas possible et l’ajouter à notre solution. On a différentes propositions et sont les suivant :

- On peut automatiser ce système pour qu’il soit chargé à faire le calcule de coefficient de présence.

- Ainsi, on peut connecter le système avec l’emploi du temps de l’école pour qu’il soit programmé automatiquement sur l’horaire de son activation et les classe.

- On peut ajouter un flash ou un signal pour qu’il indique que le système va prendre une photo et cela veut que tout élève doive être bien position en face la caméra dans ce moment.

Cas étrange :

Lorsque le système de détection ne détecte pas un visage on met l’image complète et on envoie a la personne qui est absent un mail pour qu’il justifie son absence, si un probleme de système il peut réclamer et y a l’image de toute la classe pour vérifier.

Le système prend 5 images chaques seance au minimum il faut qu’un étudiant soit présent dans deux images de la meme seance.

Si le système n’arrive pas a détecter un visage, il l’enregistre ce visage et l’image complete, et par la suite l’administration peut vérifier ce cas.

**Ajouter temps d’excution sur la machine local et sur la carte Raspberry.**

**Pour la précision ça reste la meme**

**Résultats obtenu pour le système en temps réel : (sa précision)**

**Le fonctionement actual : (genre le cahier de charge mais modifé)**

Pour la partie de développement Software, on a commencé par la recherche des différents modèles disponible et différentes solutions pour les systèmes de reconnaissance faciale. On a proposé de commencer par la collecte des images de nos camarades et au même on fait les recherches et les tests de différentes solutions.

Après qu’on a effectué des recherches approfondis, on a trouvé que l’utilisation de Deep Learning est la seule solution. Et dans cette partie on a proposé de collecter les données et de chercher une solution pour l’apprentissage profond.

Prochaine travail :

Développement d’un système automatique qui calcule les coefficients de la présence et de l’absence.

Connextion de la carte avec l’emploi de temps pour détecter exactement la salle et la classe dans la salle et pour qu’il soit programmer pour son activation.

Ressources :

<https://arxiv.org/pdf/1710.08092> --- VGG Face

<https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf> --- Cascade classifier

[https://arxiv.org/pdf/1409.1556 - VGG16](https://arxiv.org/pdf/1409.1556%20-%20VGG16)

<https://arxiv.org/pdf/1804.06655> VGGFace